ICC - *STRINGS E FUNÇÕES*

PROF. FERNANDO W CRUZ

Roteiro da aula

- TIPOS DE FUNÇÕES (VOID, ETC.)
- FUNÇÕES PARA TRATAMENTO DE STRINGS
- EXERCÍCIOS

Funções: posição das funções acessórias (i)

- Em linguagem C todo programa deve ter pelo menos a função principal **main()**.
- No entanto, programas podem ter mais de uma função, que podem vir antes ou depois da função principal

Funções colocadas depois da função **main()**

Nesse caso, o corpo da função **soma()** está depois da função **main()** e, por isso, precisa ser declarada antes da função **main()**.

```
#include <stdio.h>

int soma(int a, int b);

main () {
        int n1, n2;

        printf("\nDigite N1 :");
        scanf("%d", &n1);
        printf("\nDigite N2 :");
        scanf("%d", &n2);
        total = soma (n1, n2);
        printf("O resultado da soma é
        ", total);
} /* fim-main */

int soma(int a, int b); {
        int result;
        result = a + b;
        return(result);
} /* fim-soma */
```

Declaração da função **soma()**

Chamada da função **soma()**

Corpo da função **soma()**

Funções colocadas antes da função **main()**

Nesse caso, a função **soma()** está antes da função **main()** e, nesse caso, não precisa uma declaração separada como ocorreu no caso do *slide* anterior.

```
#include <stdio.h>

int soma(int a, int b) {
      int result;
      result = a + b;
      return(result);
} /* fim-soma */

main () {
      int n1, n2;

      printf("\nDigite N1 :");
      scanf("%d", &n1);
      printf("\nDigite N2 :");
      scanf("%d", &n2);
      total = soma (n1, n2);
      printf("O resultado da
      soma é ", total);
} /* fim-main */
```

Declaração e corpo da função **soma()**.

Chamada para a função **soma()**.

Funções que retornam valores (ii)

- As funções em geral podem retornar valores para a função chamadora (função principal **main()** ou outra).
- Nesse caso, pode-se usar o comando **return()** para gerar essa devolução de valores
- Na função que fez a chamada, deve existir uma variável associada para receber o valor que foi retornado.

Exemplo de função com retorno de valor

Nesse caso, a função **soma()** é do tipo **int**. Ou seja, retorna um valor inteiro.

```
#include <stdio.h>

int soma(int a, int b) {
      int result;
      result = a + b;
      return(result);
} /* fim-soma */

main () {
      int n1, n2, total;

      printf("\nDigite N1 :");
      scanf("%d", &n1);
      printf("\nDigite N2 :");
      scanf("%d", &n2);
      total = soma (n1, n2);
      printf("O resultado da
      soma é ", total);
} /* fim-main */
```

- Essa função recebe parâmetros por valor. Ou seja, cópias das variáveis **n1** e **n2** da função **main()** são passadas, respectivamente, para as variáveis **a** e **b**.
- Essa função retorna um valor (**result**) para a função chamante que é copiado (passado por valor) para a variável **total** da função main().
- Perceba que a função **soma()** é do tipo **int**, que é o mesmo tipo da variável **result** que está sendo retornada.

Chamada para a função **soma()**. Aqui, a variável **total** vai ficar com o valor retornado pela função **soma()**.

Funções que NÃO retornam valores (iii)

- As funções que não retornam valores não precisam do comando **return()**, como no caso anterior
- Geralmente essas funções trabalham com variáveis globais (declaradas fora de qualquer função) ou recebem parâmetros por referência
- Na função que fez a chamada não precisa existir uma variável associada para receber valores. Veja exemplo no *slide* seguinte
- Esse tipo de função pode ser declarada como sendo do tipo **void**, que quer dizer: função "sem tipo ou sem parâmetro definido de retorno".

Exemplo de função sem tipo ou do tipo **void** (sem retorno de parâmetro)

```c
#include <stdio.h>
#define TAM_MAX 50

void leia_nome(char s[]) {
  int i=0, c;

  c = getchar();
  while (c!='\n') {
     s[i] = c;
     c = getchar();
     i++;
  } /* fim-while */
  s[i]   = '\n';
  s[i+1] = '\0';
} /* fim-leia_nome */


main () {
  char aluno[TAM_MAX];

  printf("\nDigite o nome do aluno : ");
  leia_nome(aluno);
  printf("\nO nome digitado foi    : %s\n", aluno);
} /* fim-programa */
```

- Essa função recebe parâmetro por referência. Ou seja, a variável **s** endereça a mesma posição de memória que a variável **aluno** da função **main()** chamadora.
- Perceba que os valores digitados para a variável **s** equivalem à variável **aluno** da função **main()**
- Essa função é do tipo **void**, ou seja, não tem um tipo definido para ela
- Perceba que essa função não tem o comando **return()**.

Chamada para a função **leia_nome()**. Perceba que não existe uma variável para receber os valores dessa função, como ocorreu no exemplo anterior.

STRING(s): relembrando as funções *getchar()* e *putchar()*

FUNÇÃO ***getchar( )***

Lê um caractere da console após pressionada a tecla "ENTER", usando **buffer**, e retorna '-1' se nada foi lido.

Exemplo:

```
:
c = getchar( );
:
```

FUNÇÃO ***putchar()***

Escreve o valor de um único caracter passado pelo argumento da função, que pode ser uma outra função.

Exemplo:

```
:
c = getchar( );
putchar(c);
:
```

ou como

```
:
putchar(getchar());
:
```

Impressão de *strings*

- *Strings* podem ser impressas com as funções **putchar()** e **printf()**
- Máscaras de impressão de *strings* para o **printf()**:
  - %s = imprime uma *string* inteira
  - %c = imprime um caracter por vez
- A função **putchar()** não precisa de máscara de impressão
- Veja exemplos no *slide* a seguir:

Impressão de *string* usando **printf()** e máscara **%s**

```c
#include <stdio.h>
#define TAM_MAX 50

void leia_nome(char s[]) {
  int i=0, c;

  c = getchar();
  while (c!='\n') {
     s[i] = c;
     c = getchar();
     i++;
  } /* fim-while */
  s[i]   = '\n';
  s[i+1] = '\0';
} /* fim-leia_nome */


main () {
  char aluno[TAM_MAX];

  printf("\nDigite o nome do aluno : ");
  leia_nome(aluno);
  printf("\nO nome digitado foi    : %s\n", aluno);
} /* fim-programa */
```

Impressão com **printf()** e máscara **%s.** Com a opção %s, o printf imprime todos os caracteres até encontrar o '\0' que foi colocado pela função leia_nome (). Ou seja, existe um laço implícito de impressão, caracter a caracter. Compare essa impressão com opção **%c** que está no próximo *slide*.

Impressão de *string* usando **printf()** e máscara **%c**

```c
#include <stdio.h>
#define TAM_MAX 50

void leia_nome(char s[]) {
   int i=0, c;

   c = getchar();
   while (c!='\n') {
       s[i] = c;
       c = getchar();
       i++;
   } /* fim-while */
   s[i]   = '\n';
   s[i+1] = '\0';
} /* fim-leia_nome */

main () {
   char aluno[TAM_MAX];
   int i=0;
   printf("\nDigite o nome do aluno : ");
   leia_nome(aluno);
   printf("\nO nome digitado foi    : ");
   i = 0;
   while (aluno[i]!='\0') {
       printf("%c", aluno[i]);
       i++;
   } /* fim-while */
} /* fim-programa */
```

Impressão com **printf()** e máscara **%c.** Perceba que aqui o controle de parada é feito pelo '\0' colocado no fim da variável *string* **nome** (veja função leia_nome)

Impressão de *string* usando a função **putchar()**

```c
#include <stdio.h>
#define TAM_MAX 50

void leia_nome(char s[]) {
   int i=0, c;

   c = getchar();
   while (c!='\n') {
       s[i] = c;
       c = getchar();
       i++;
   } /* fim-while */
   s[i]   = '\n';
   s[i+1] = '\0';
} /* fim-leia_nome */

main () {
   char aluno[TAM_MAX];
   int i=0;
   printf("\nDigite o nome do aluno : ");
   leia_nome(aluno);
   printf("\nO nome digitado foi    : ");
   i = 0;
   while (aluno[i]!='\0') {
       putchar(aluno[i]);
       i++;
   } /* fim-while */
} /* fim-programa */
```

Impressão com **putchar().** Esse comando não necessita de máscara e imprime apenas um caracter por vez. Perceba que aqui o controle de parada é feito pelo '\0' colocado no fim da variável *string* **nome** (veja função leia_nome)

Exercício 1: Dado o programa abaixo, construir as funções **leia_nome** e **leia_nota**.

```c
#include <stdio.h>
#define TAM_MAX 50
#define TOTALUNOS 4

void leia_nome(char s[]);
int leia_nota(int aluno);

main () {
   char    naluno[TOTALUNOS][TAM_MAX];
   int     notaluno[TOTALUNOS];
   int     i = 0, ind=-1, maior_nota;

   system("cls");
   while (i<TOTALUNOS) {
       printf("----------------------");
       printf("\nSobre o aluno %d\n", i);
       leia_nome(naluno[i]);
       notaluno[i] = leia_nota(i);
       i++;
   } /* fim-while */

   maior_nota = -1;
   i = 0;
   while (i<TOTALUNOS) {
       if (notaluno[i]>maior_nota) {
           maior_nota = notaluno[i];
           ind = i;
       } /* fim-if */
       i++;
   } /* fim-while */
   printf("\n---------------------------------------\n");
   printf("LAUREADO : %s COM NOTA %d", naluno[ind], notaluno[ind]);
   printf("\n---------------------------------------\n");
} /* fim-programa */
```

Definição de constantes do programa

Declaração de funções a serem usadas no programa. Nesse caso, são descritas depois da funcão *main*

Chamada às funções. Perceba que a função leia_nome não retorna valor e, por isso, é do tipo na sua declaração ***void***.

Essa parte do programa descobre o aluno que tem maior nota e imprime o seu nome e nota

Solução do exercício 1: Função leia_nome()

```
void leia_nome(char s[]){
  int c;
  int i=0;

  printf("NOME : ");
  c=getchar();
  if (c=='\n') {
     c=getchar();
  } /* fim-if */
  while (c!='\n') {
     s[i] = c;
     i = i+1;
     c = getchar();
  } /* fim-while */
  s[i]   = '\n';
  s[i+1] = '\0';
} /* fim-leia_nome */
```

O vetor **s** aqui refere-se a uma das linhas da matriz **naluno** da função **main**. Por exemplo, ao chamar leia_nome(naluno[0]), o vetor **s** está relacionado à linha zero da matriz original; ao chamar leia_nome(naluno[1]), o vetor **s** passa a referenciar a a linha um da matriz original, e assim por diante.

Normalmente bastaria um único getchar. No entanto esse teste com elimina possíveis Line Feeds (caracter '\n') que ficaram armazenados em memória de teclado.

Prepara o vetor para ser utilizado pelas funções típicas de manipulação de strings (printf, etc.)

Solução do exercício 1: Função leia_nota()

```
int leia_nota(int aluno) {
  int nota;

  printf("NOTA : ");
  scanf("%d", &nota);
  return(nota);
} /* fim-leia_nota */
```

Exercícios extras

1) Fazer um algoritmo para converter temperaturas de graus Celsius para Fahrenheit. No programa, a função principal (main) deve solicitar valores de temperatura em Celsius e imprimir o equivalente em Fahrenheit até que o usuário digite uma temperatura igual a "9999". O processo de conversão para Fahrenheit deve ocorrer dentro de uma função.

Exercícios extras

2) Fazer um programa para calcular o valor de S para os N primeiros termos da série abaixo.

$$S = 1!/2^3 + 3!/4^6 + 5!/6^9 + 7!/8^{12} + ...$$

No cálculo do fatorial e da exponenciação, criar as funções fatorial e power conforme a seguinte especificação:

- y = fatorial (n), onde fatorial calcula e retorna o fatorial de ***n***; ***y*** = inteiro longo e ***n*** é inteiro.
- y' = power (n, e), onde power é a função que calcula o resultado da operação de ***n*** elevado ao expoente ***e***; n e ***e*** são números inteiros positivos; ***y'*** = inteiro longo.

Exercícios extras

3) Fazer um programa para ler os nomes de um grupo de 50 alunos e determinar qual deles tem o maior nome. Requisitos do programa:

i) Para a leitura do nome, usar o getchar() dentro de uma função específica para essa leitura. Essa função deve retornar o tamanho do nome lido.

ii) Criar a função copia(char r[], char [s]) que gera uma cópia da *string* **r** para a *string* **s**. Essa função será chamada todas as vezes que o nome lido for maior que o maior nome atual.